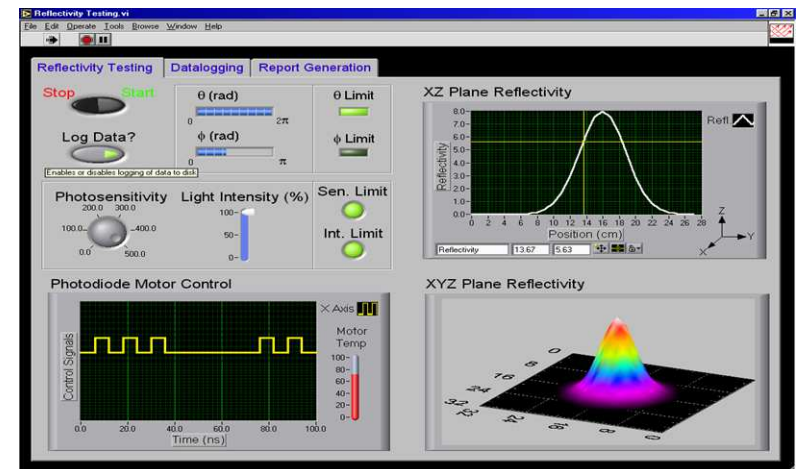
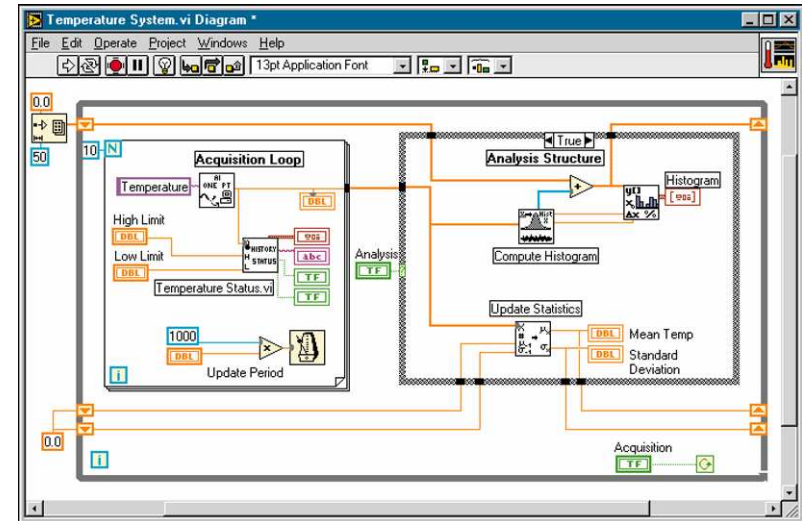


# EPICS and LabVIEW

Tony Vento, National Instruments  
Willem Blokland, ORNL-SNS



- Graphical dataflow programming
- Interactive front panel / GUI
- Efficient compiled execution
- Targets
  - Windows, Real-Time, FPGA, Linux, Macintosh, DSP, Other Processors
- I/O and analysis libraries
- Distributed networking capabilities



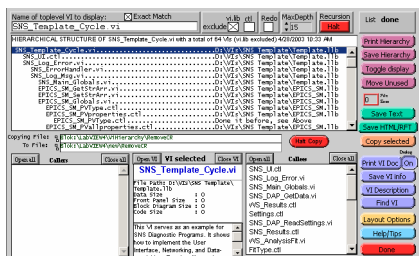
# EPICS and LabVIEW Interfaces

## (Oak Ridge and Others)

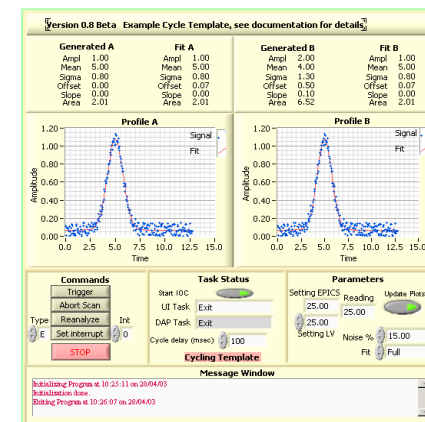
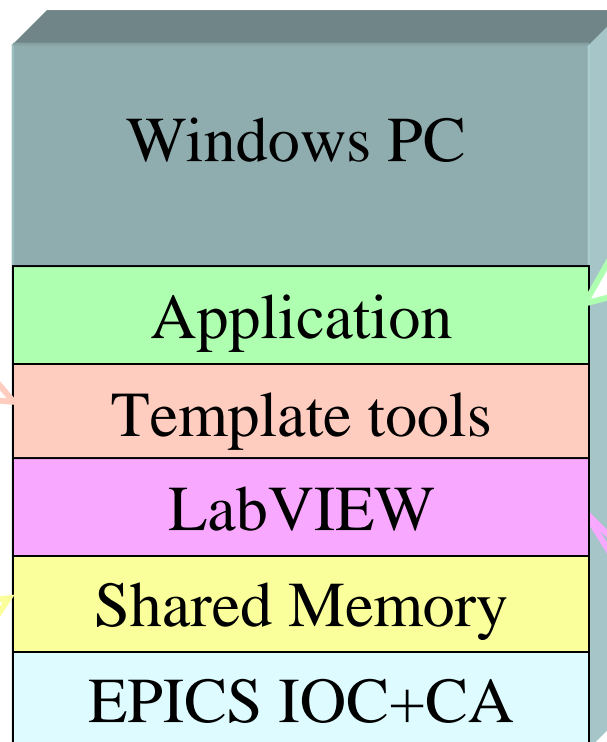
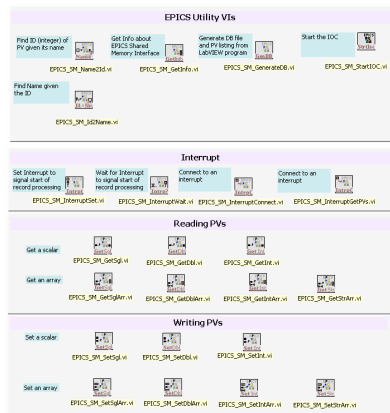
1. Shared Memory Interface to IOC (I/O Channel)
  - Links LabVIEW and IOC Process Variables (PVs)
  - Data from LabVIEW is available to the IOC
  - Windows
2. CA (Channel Access) Client
  - LabVIEW as a display environment for PVs
  - No programming required
  - Windows, Macintosh, Linux

# 1. Shared Memory Interface

- db & cmd file generator at startup
- Development tool -Cloning/documentation



- Shared Memory Records
- LabVIEW Library (D. Thompson)



- Standard Examples
- NADs implemented: BCM, BPM, ES, FC, TTS (M. Sundaram, C. Long, W. Blokland, LANL, BNL)

- Graphical programming
- Drivers to DAQ
- Processing routines
- Graphs/Plots

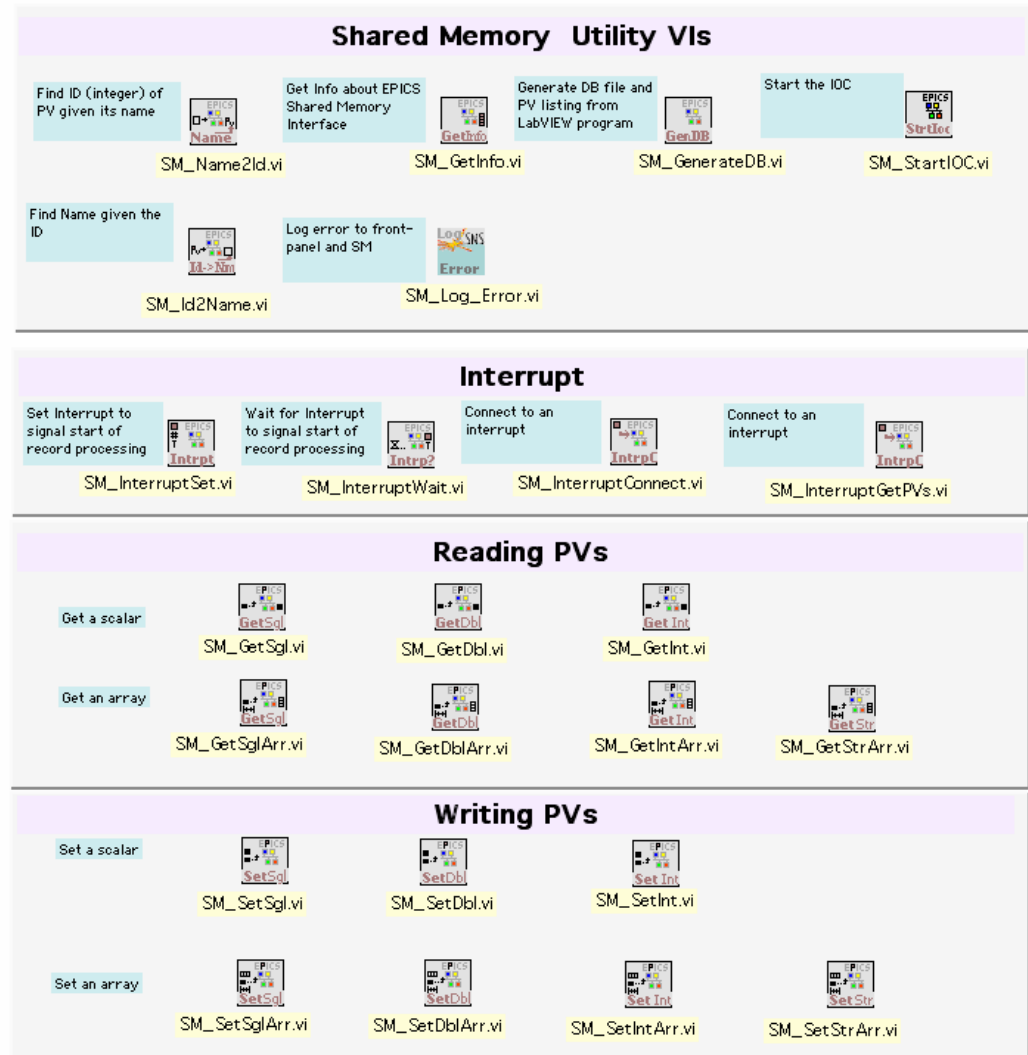
Accelerator Controls

# Shared Memory Interface

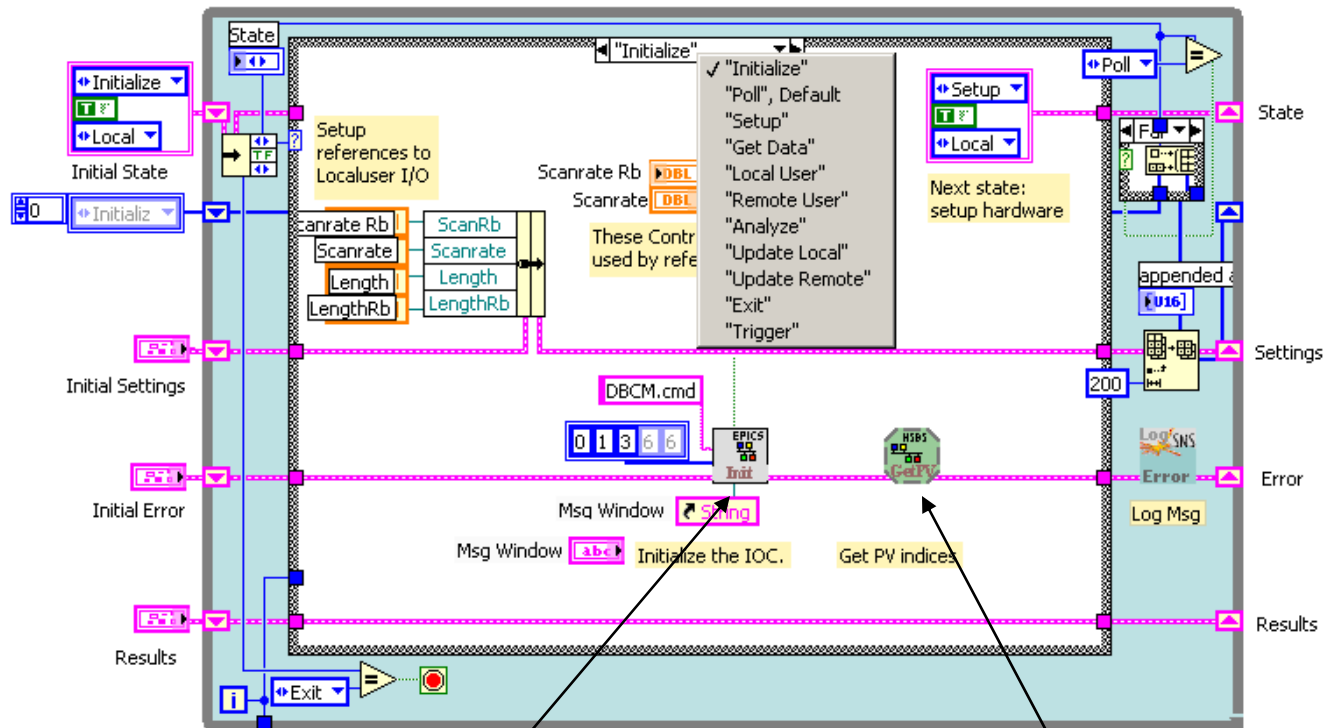
Shared Memory IOC written by D. Thompson. LabVIEW Library by W. Blokland.

The shared memory interface buffers the data and implements functions to:

- **Create, find and destroy variables**
- **Read from and write to variables**
- **Set and receive events**
- **Retrieve information about variables**



# Program startup



1. LabVIEW loads and runs Top-Level VI
2. LabVIEW gets variable declarations, generates .db and starts IOC
3. IOC starts and loads db file. Each time the IOC creates a PV record, the record creates a Shared Memory (SM) entry
4. LabVIEW finds references to SM entries by resolving PV names

# EPICS dB Utility

The screenshot shows the EPICS dB Utility interface with the following sections:

- System\_Subsystem:** A text box containing "Example\_Basic".
- Project:** A text box containing "Basic3".
- Accelerator System:** A table with two columns: "Variable" and "Value". The first row is highlighted in blue.
- error in (no error):** A section with a green checkmark icon, a "status" field with the value "d0", and a "source" field.
- .db File:** A text area containing the following code:

```
record(ai, "${SS}:${A}:ADC")
{
  field(DTYP, "SMDouble")
  field(DESC, "${A}ADC")
  field(INP, "@${A}ADC")
  field(SCAN, "1 second")
  field(PINI, "YES")
  field(TSE, "-2")
}
```
- .cmd File:** A text area containing the following code:

```
dbLoadDatabase("../dbd/epicsSM.dbd",0,0)
registerRecordDeviceDriver(pdbbase)
dbLoadRecords("../db/Basic3.db", "SS=Example_Basic,
A=Basic3")
iocInit()
#seq sncExample, "SS=Example_Basic, A=Basic3"
```
- .xls file:** A table with columns: "Name", "Type", "In/Out", "#Elements", "Scan", "Hz", "Interrupt". It lists four records for "Example\_Basic:Basic3".

Name	Type	In/Out	#Elements	Scan	Hz	Interrupt
Example_Basic:Basic3:ADC	Double	In	1	Scan	1.000000	-1
Example_Basic:Basic3:Coefficient	Double	Out	1	Passive	1.000000	-1
Example_Basic:Basic3:CoefficientRb	Double	In	1	Scan	1.000000	-1
Example_Basic:Basic3:DAC	Double	Out	1	Passive	1.000000	-1
Example_Basic:Basic3:DACRb	Double	In	1	Scan	1.000000	-1

The dBGenerate Utility automatically generates the database file for the EPICS IOC as well as the cmd file and an Excel file. The utility uses the VI that resolves names to indices, as written by the programmer, as information on which PVs are going to be used. The utility supports the instantiation of one device many times (necessary for the NAD concept).

# Performance

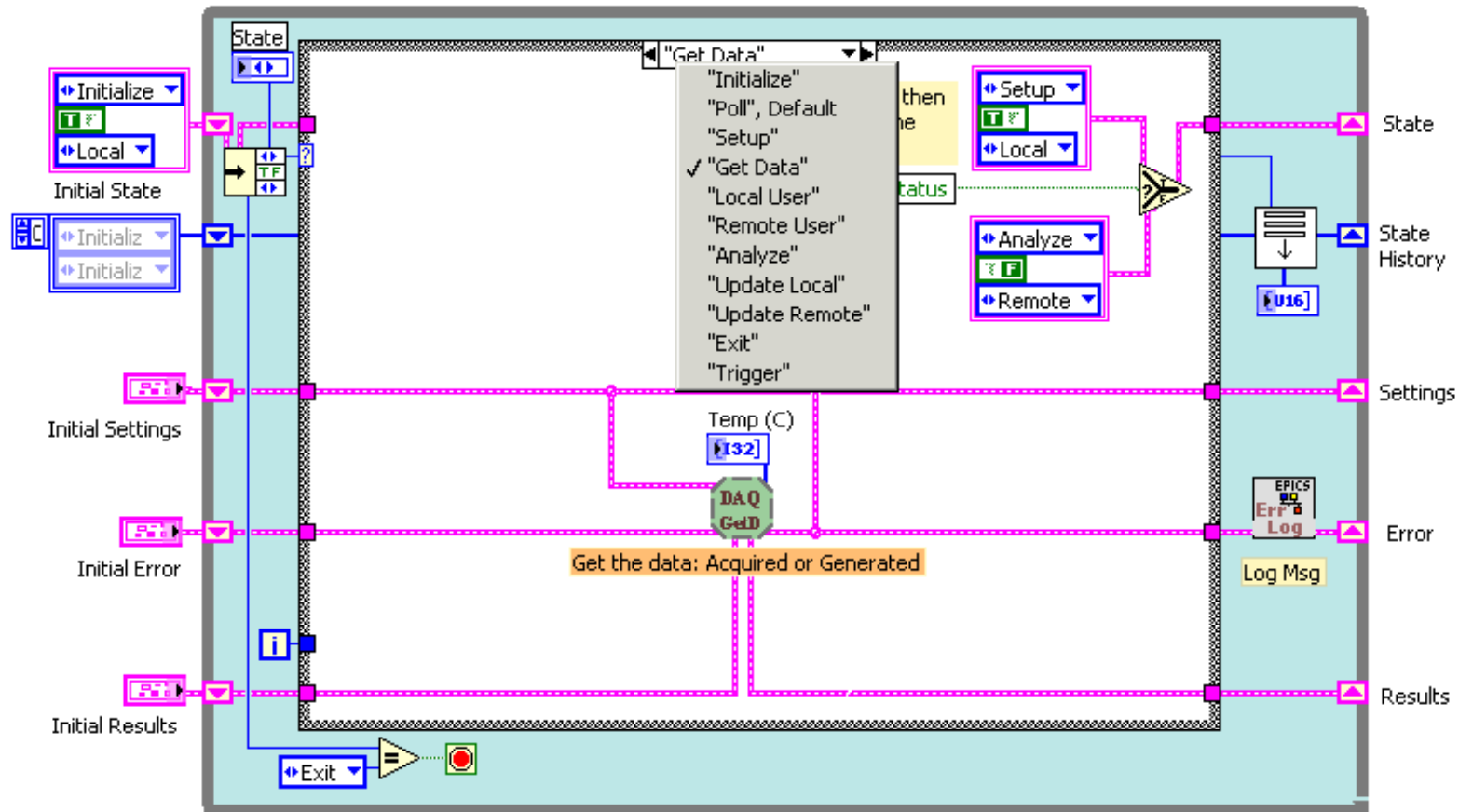
TEST	Profiler	Total program	Comments
Writing 4096 doubles and setting interrupt to process values	80 usec (+- 10 usec) + 20 usec (+- 5 usec)	270 usec	(2Ghz P4) no client
Writing 1024 doubles and setting interrupt to process values	67 + 11	130 usec (+-10usec)	

Profiler: times measured with LabVIEW's profiler tool, excludes all time spent by processor elsewhere

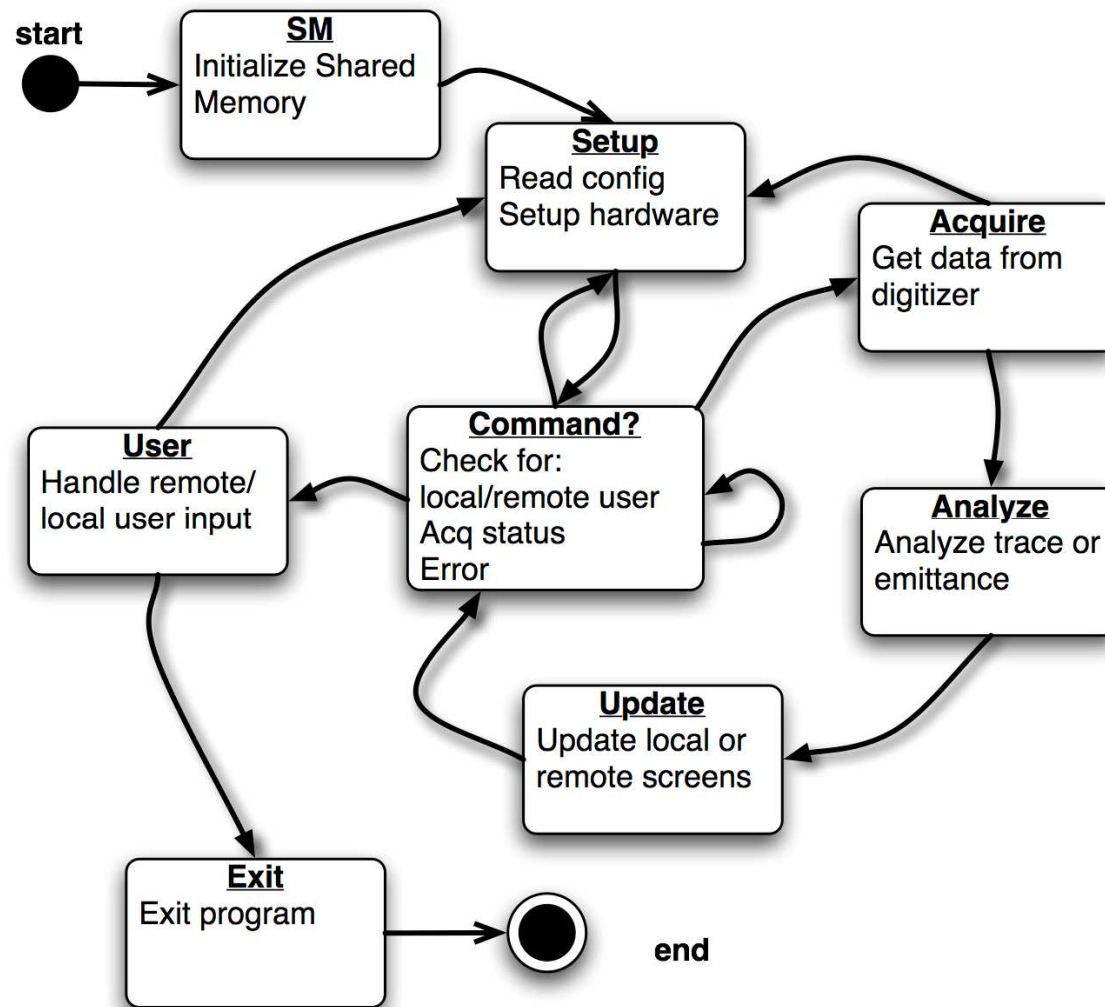
Total: use time before and after execution, includes all time spent by processor on other processes.



# Template: State Machine



# Template: State Machine diagram



# Template

Use templates (state-machine based) to:

- 1) Strict organization -> clear development path
- 2) Similar structure among instruments-> re-use of code with copy/rename tool
- 3) Simple structure -> fast development and debug
- 4) Built-in support for user-interface/EPICS/ configuration files.
- 5) State machine supports jumps to other state if exception occurs (e.g. error condition)

Network Attached Device: Each pickup or sensor has its own resources such as timing, data acquisition and processing. No tightly coupled systems An individual device can fail or be serviced without disrupting other devices. Software is simplified as it doesn't have to manage single resources among multiple devices.

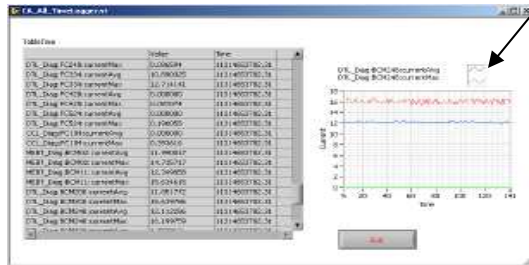
# Diagnostic NADs

## (Network Attached Device)

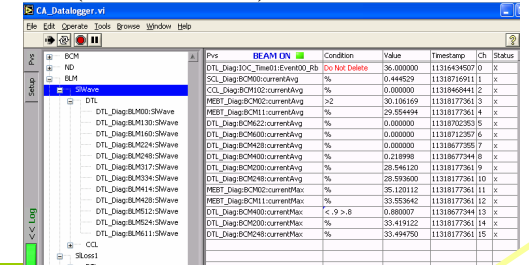
Instrument	Type	Comments	Programmer
Beam Position Linac	IOC	Custom	Blokland/LANL
Beam Position Ring	IOC	Custom	Long/BNL/Blokland
Wire Scanner	CA	Commercial with custom pre-amps	Chistensen (Blokland)
Beam Stop	IOC	Commercial	Blokland
Beam Aperture	IOC	Commercial	Blokland
Faraday Cup	IOC	Commercial	Blokland
Arc Detector	IOC	Commercial	Armstrong/Nesterenko
Source Current Logger	IOC	Commercial	Stedinger
Beam Current Monitor (Linac/Ring)	IOC	Commercial with filter box	Blokland/ArmstrongBNL
Faraday Cup with energy degrader	IOC	Commercial	Blokland/LANL
Halo Scraper/ Beam Stop	IOC	Commercial	Blokland/LANL
Fast Beam Loss Proto	IOC	Commercial	Blokland
Bunch Shape Monitor	(IOC)		INR
Emittance	IOC	Commercial	Long/Blokland
Neutron Detector	IOC	Commercial	Liyu
Physics Test Node	IOC	Commercial	Blokland
Harp	IOC	Commercial	Sundaram
Laserwire transferline	IOC	Commercial	Long
Laserwire beambox	IOC	Commercial	Long
Laserwire video	IOC	Commercial	Nesterenko/Blokland
Video Foil	IOC	Commercial	Armstrong/Blokland/BNL

## 2. CA (Channel Access) Client

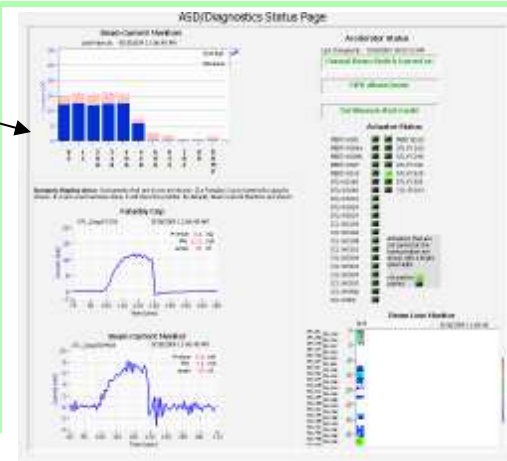
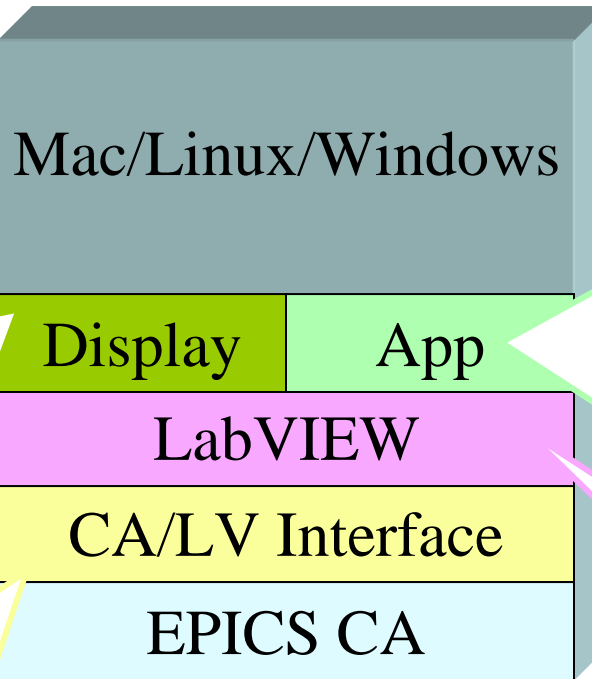
- Drag and Drop Display Client for implementing screens that don't require programming (W. Blokland)



- Tools to select data to log or set (M. Sundaram)



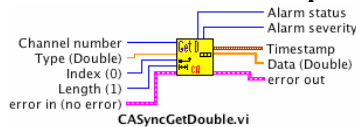
Processing the data.



Examples:

- Channel13: dynamically loaded display modules
- NAD Control screens (W. Blokland, D. Purcell, M. Sundaram)

Multi-platform Interface to buffer CA messages and handle interrupts (A. Liyu)



Channel Access

Graphical programming

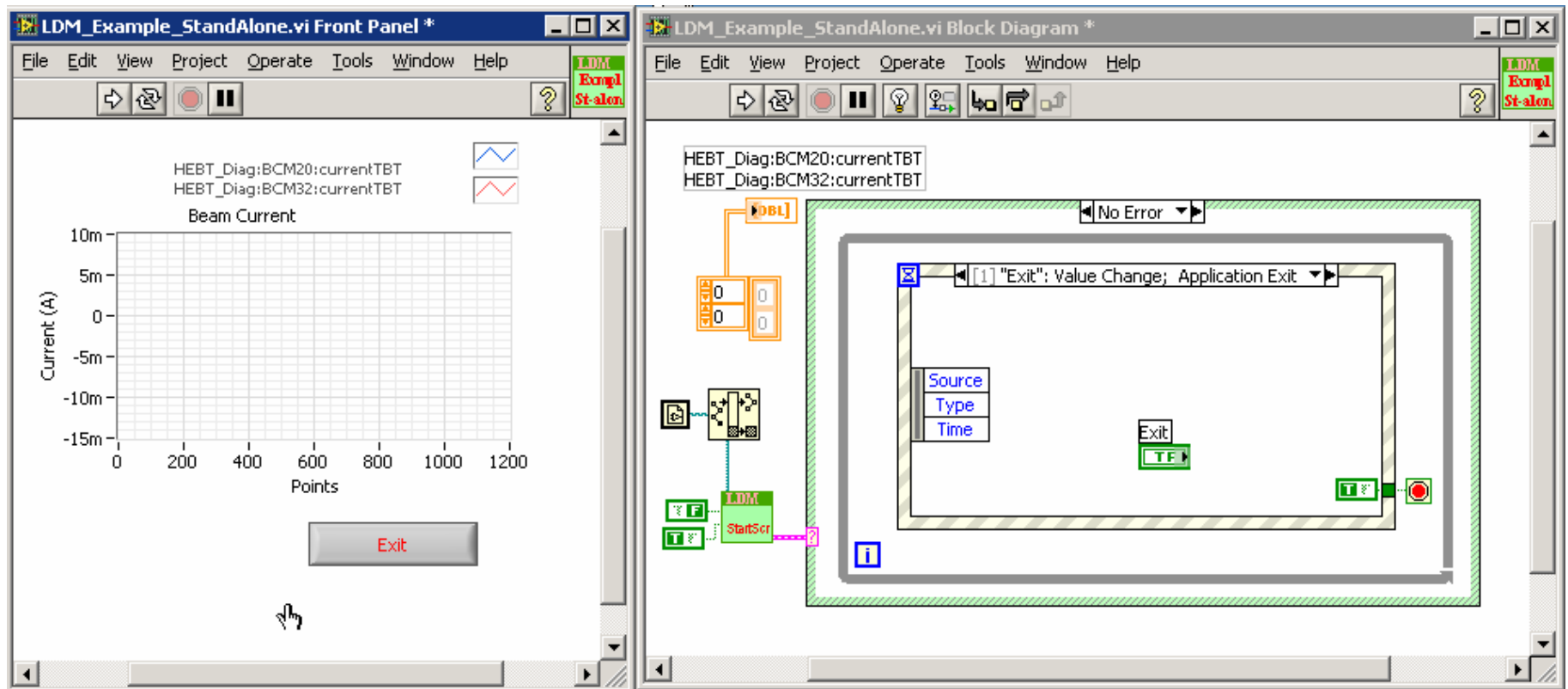
- Processing routines
- Graphs/Plots

Accelerator Controls

# Automatic PV Generation

- Adding a PV is simple, just add a control to LabVIEW front panel, and then edit one field.
- New method generates PVs automatically at runtime.
- Adding a PV now takes around 2 minutes on average.
- Not prone to error.
- Currently supports scalar numbers, one dimensional arrays of numbers, and it will parse down into clusters.

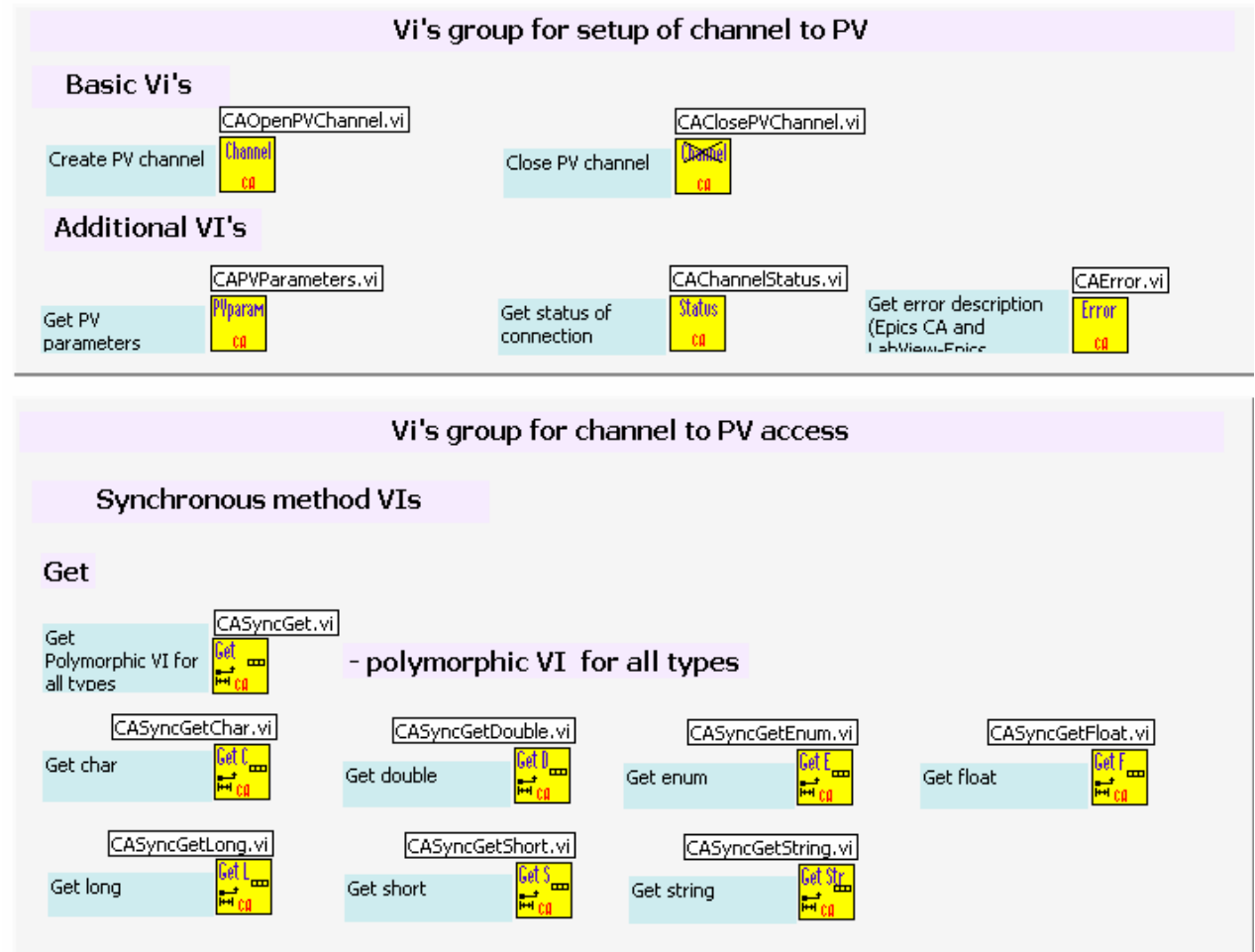
# LabVIEW Display Manager



- Name a graph, control, scalar display after a PV then run.
- Analysis can be added to the code
- Implementation: LabVIEW can ask about its program, e.g. what are the parts on my panel, refer to these parts and send data to these parts

# LabVIEW Channel Access Client

Developed by  
Andrei Liyu to  
Access PVs.  
(monitoring,  
scan, set)

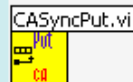




# LabVIEW Channel Access Client

## Put

Put.  
Polymorphic VI for  
all types



- polymorphic VI for all types

CASyncPutChar.vi

Put char



CASyncPutDouble.vi

Put double



CASyncPutEnum.vi

Put enum



CASyncPutFloat.vi

Put float



CASyncPutLong.vi

Put long



CASyncPutShort.vi

Put short



CASyncPutString.vi

Put string



## Monitor method VIs

Set monitor



Unset monitor



Callback  
Polymorphic VI for  
all types



- polymorphic VI for all types

CACallbackChar.vi

Callback char



CACallbackDouble.vi

Callback double



CACallbackEnum.vi

Callback enum



CACallbackFloat.vi

Callback float



CACallbackLong.vi

Callback long



CACallbackShort.vi

Callback short

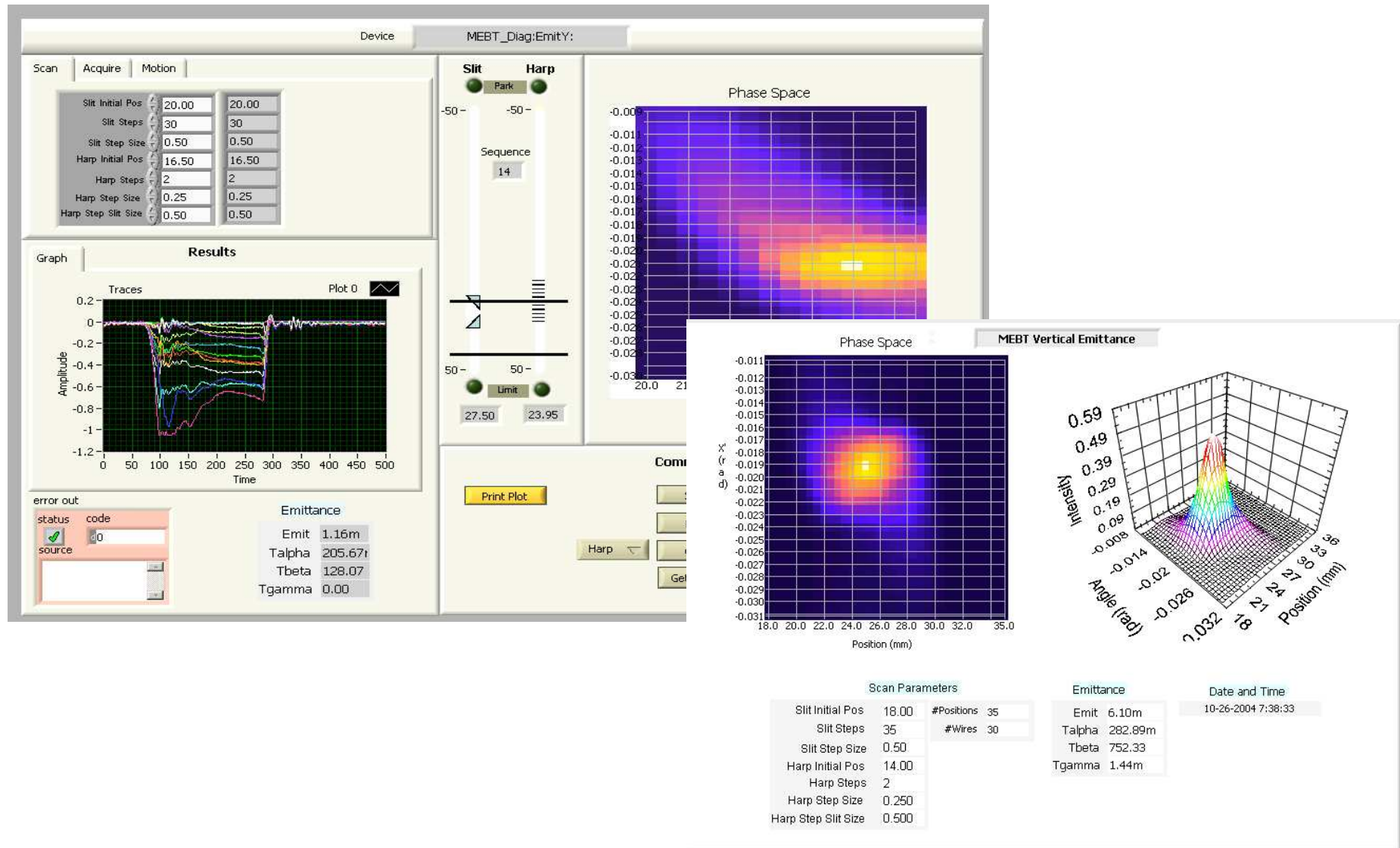


CACallbackString.vi

Callback string



# Emittance scanner CA Client



# LabVIEW/EPICS on the Web

The screenshot shows the SNS LabVIEW Shared Memory Interface to EPICS IOC web page. The header features the SNS logo and a map of the United States with the text "A U.S. Department of Energy multilaboratory project Spallation Neutron Source". The left sidebar contains navigation links: SM IOC, CA Client, Contacts, Diagnostics, Controls, EPICS, SNS Home, Project Site, and ORNL. The main content area is titled "LabVIEW Shared Memory Interface to EPICS IOC" and contains a description of the interface, a list of features, and two screenshots of the LabVIEW interface.

**LabVIEW Shared Memory Interface to EPICS IOC**

The Shared Memory Interface links LabVIEW variables to EPICS IOC Process Variables (PVs). Data acquired and processed by LabVIEW is available to the IOC to communicate to an EPICS based control system. LabVIEW and the IOC can also send interrupts/signals to notify each other that data is available.

Features:

- LabVIEW starts the IOC with a command file to load a .db file for the application. This is a .db file generated by a LabVIEW utility and derived from the LabVIEW code that refers to the PVs (you don't have to know db syntax unless you want to add to the file).
- Current version is based on EPICS 3.14beta, Windows only
- Library of VIs accessing the Shared Memory Interface and creating .db files
- Template VIs and Style Guide to demonstrate use of LabVIEW. Event driven and Polling examples.
- Development tool to document LabVIEW programs, clone projects, and manage VIs.

The bottom of the page shows two screenshots of the LabVIEW interface. The left screenshot displays a graph of a signal over time, with a peak and a dip. The right screenshot shows a LabVIEW block diagram with various components, including a graph, a table, and a message window.

Many people from labs  
have requested the SNS  
LabVIEW/EPICS interface  
(France, Germany, Italy, Korea, U.S.,  
China, etc)

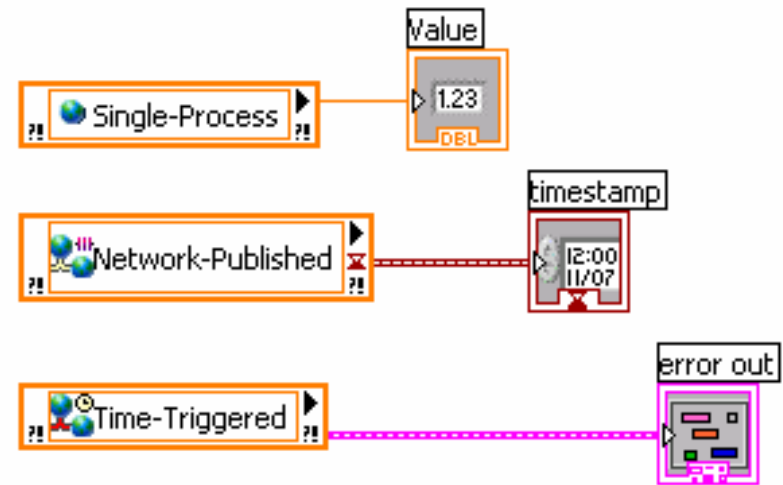
[http://www.sns.gov/diagnostics/documents/epics/LabVIEW/SNS\\_LabVIEWEPICS.html#Top](http://www.sns.gov/diagnostics/documents/epics/LabVIEW/SNS_LabVIEWEPICS.html#Top)

<http://www.sns.gov/diagnostics/documents/epics/Downloads/downloads.html>

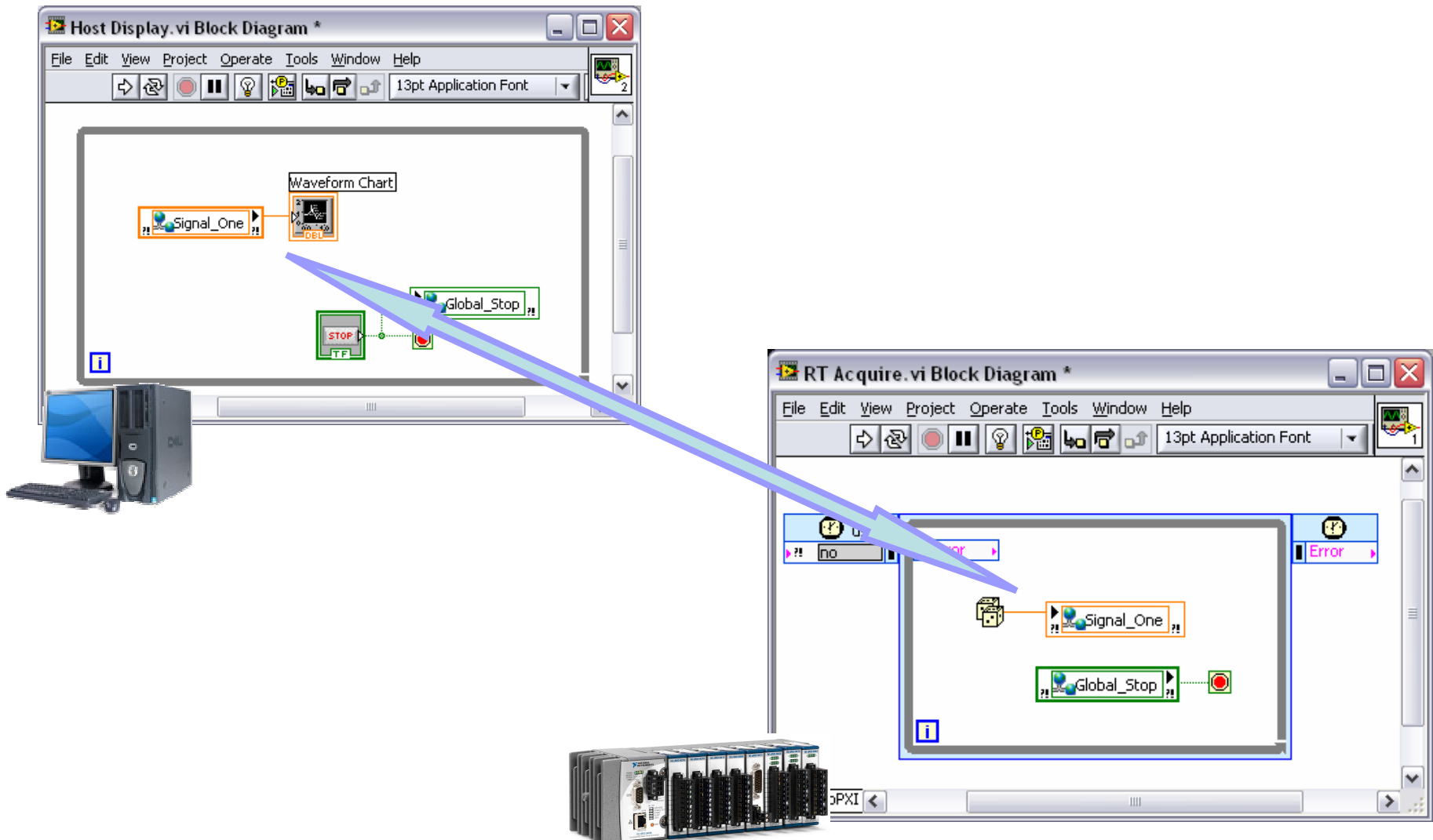
Looking Forward

# Shared Variable in LabVIEW 8

- Single-Process
  - Use instead of Global Variable
- Network-Published
  - Available to any node
- Time-Triggered
  - Deterministic Communication



# Communication between Targets



# I/O Server in LabVIEW 8

- An I/O Server is a plug-in to the shared variable engine
- An I/O Server could be created for the CA (Channel Access) Protocol